What is claimed is:

1. A method for minimizing compatibility issues among interacting components of different dialect versions, comprising:

defining a plurality of type-version identifiers each indicating a corresponding type and version from among a plurality of types and versions of requests;

installing at least one handler at each of a first component and a second component, each of said handlers supporting a corresponding one of said plurality of types and versions of requests;

sending a request from the first component to the second component, said sent request having a particular identifier that is one of the plurality of type-version identifiers indicating a particular type and version of said sent request; and

causing said second component to,

extract said particular identifier of said sent request,

use said particular identifier to determine whether one of said handlers installed at said second component properly supports said particular type and version of said sent request, and

if a proper one of said installed handlers supports said particular type and version of said sent request, using said proper handler to process said sent request.

2. A method as recited in claim 1, further comprising:

providing said first component with a first data structure for indicating whether or not said first component may send requests of said particular type and version to said second component; and

before said step of sending a request, accessing said first data structure to determine whether requests of said particular type and version may be sent to said second component.

3. A method as recited in claim 2, further comprising:

reserving at least one of said plurality of type-version identifiers for indicating a stop sending control request, said stop sending control request having a first field carrying said reserved type-version identifier, and a second field carrying a second one of the plurality of type-version identifiers; and

if none of said installed handlers at said second component properly supports said particular type and version of said sent request, causing said second component to send a stop sending control request to said first component to indicate that said first component should stop sending requests of said particular type and version to said second component.

4. A method as recited in claim 3, further comprising:

receiving said stop sending control request at said first component; and

updating said first data structure to indicate that said first component may not send requests of said particular type and version to said second component.

5. A method as recited in claim 1, wherein said sent request carries said particular identifier in a header field, the method further comprising:

extending said particular identifier by defining a sub-type-version identifier in said header field, said particular identifier having a value that indicates the presence of said sub-type-version identifier in said header field.

6. A method as recited in claim 5, the method further comprising:

extending said sub-type-version identifier by defining a sub-sub-type-version identifier in said header field, said sub-type-version identifier having a value that indicates the presence of said sub-sub-type-version identifier in said header field.

7. A method as recited in claim 1, wherein said step of using said particular identifier to determine whether one of said handlers installed at said second component properly supports said particular type and version of said sent request further comprises:

using said particular identifier to index a second data structure having a plurality of pointers, each said pointer being associated with one of said plurality of type-version identifiers and pointing to a corresponding one of said handlers installed at said second component.

8. A method as recited in claim 7, further comprising:

installing an unsupported-type-version handler at said second component, said unsupported-type-version handler supporting requests carrying any one of a plurality of unsupported ones of said type-version identifiers wherein said sent request carries one of said unsupported type-version identifiers; and

invoking said unsupported-type-version handler in response to said received unsupported type-version identifier without indexing said second data structure;

whereby said second data structure need not store pointers for each of said unsupported type-version identifiers.

9. A versioning infrastructure for minimizing compatibility issues among interacting components of different dialect versions, comprising:

a plurality of components between which requests may be exchanged, each request being of a type and version from among a plurality of types and versions and having a header carrying a type-version identifier indicating a corresponding type and version of said request; and

each said component including,

an input port for receiving one of said requests,

at least one handler supporting requests of a corresponding one of the plurality of types and versions,

a pointer array having a plurality of elements each being a pointer to a corresponding one of said handlers, and

switching logic operable to

extract said type-version identifier carried by said received request,

use said extracted type-version identifier to index said pointer array to determine a selected one of said handlers, and

invoke said selected handler.

10. A versioning infrastructure as recited in claim 9, wherein each said component further includes installation logic operable to install said handlers at said component.

11. A versioning infrastructure as recited in claim 9, wherein each said component further includes a flag array having at least one flag element corresponding to an intended receiver component and to a particular one of the plurality of types and versions, each said flag element indicating whether or not said component may send requests of said corresponding particular type and version to said corresponding intended receiver component.

12. A versioning infrastructure as recited in claim 9, wherein each said component further includes incompatibility reporting logic operable to report receipt of a request that is not supported by any one of said installed handlers.

13. A versioning infrastructure as recited in claim 9, wherein said requests include at least one control request for managing versioning in accordance with the infrastructure, each said control request carrying a corresponding control type-version identifier specifying a type of control request, said control requests including a stop sending control request to be sent by a notifying one of the components to a receiving one of the components to indicate that said receiving component should stop sending requests of a particular one of the plurality of types and versions to said notifying component.

14. A versioning infrastructure as recited in claim 13, wherein said control requests further include a start sending control request to be sent by a notifying one of the components to a receiving one of the components to indicate that said receiving component may start sending requests of a particular one of the plurality of types and versions to said notifying component.

15. A versioning infrastructure as recited in claim 13, wherein said control requests further include a test connection request to be sent by a notifying one of the components to a receiving one of the components to indicate that said receiving component may send a test connection response to said notifying component so as to probe an underlying connection between said notifying component and said receiving component.

16. A versioning infrastructure as recited in claim 13, wherein said control requests further include a message reporting request to be sent by a notifying one of the components that has

received an unrecognized request to a receiving one of the components to indicate that said receiving component may send a message reporting response that has a message describing said unrecognized request to said notifying component.

17. A versioning infrastructure as recited in claim 9, wherein said header of each said request carries a sub-type-version identifier for extending said type-version identifier, said type-version identifier having a value that indicates the presence of said sub-type-version identifier.

18. A versioning infrastructure as recited in claim 17, wherein said header of each said request carries a sub-sub-type-version identifier for extending said sub-type-version identifier, said sub-type-version identifier having a value that indicates the presence of said sub-sub-type-version identifier.

19. A versioning infrastructure as recited in claim 9, wherein

one of said handlers is an unsupported-type-version handler, said unsupported-type-version handler supporting requests carrying any one of a plurality of unsupported ones of said type-version identifiers; and

said switching logic includes memory saving logic that invokes said unsupported-type-version handler in response to one of said unsupported-type-version identifiers carried by said received request without indexing said pointer array;

whereby said pointer array need not store pointers for each of said unsupported type-version identifiers.

20. A versioning infrastructure for minimizing compatibility issues among interacting components of different dialect versions, comprising:

a plurality of components between which requests may be exchanged, each request being of a type and version from among a plurality of types and versions and having a header carrying a type-version identifier indicating a corresponding type and version of said request; and

each said component including,

means for receiving one of said requests,

means for installing at least one handler at said component, each said handler supporting a corresponding one of the plurality of types and versions,

data structure means having a plurality of elements each being a pointer to a corresponding one of said handlers,

means for extracting said type-version identifier carried by said received request,

means for using said extracted type-version identifier to index said data structure means to determine a selected one of said handlers, and

means for invoking said selected handler.

21. Computer readable medium embodying program code with instructions for minimizing compatibility issues among interacting components of different dialect versions, comprising:

program code to cause a computer to define a plurality of type-version identifiers each indicating a corresponding type and version from among a plurality of types and versions of requests;

program code to cause a computer to install at least one handler at each of a first component and a second component, each of said handlers supporting a corresponding one of said plurality of types and versions of requests;

program code to cause a computer to send a request from the first component to the second component, said sent request having a particular identifier that is one of the plurality of type-version identifiers indicating a particular type and version of said sent request: and

program code to cause a computer having the said second component to:

extract said particular identifier of said sent request,

use said particular identifier to determine whether one of said handlers installed at said second component properly supports said particular type and version of said sent request,

if a proper one of said installed handlers supports said particular type and version of said sent request, using said proper handler to process said sent request.

22. A version identifier as recited in any of claims 1, 9, 20, or 21, wherein the dialect is a protocol.

23. A version identifier as recited in any of claims 1, 9, 20, or 21, wherein the dialect is an interface.

24. A communication network including a versioning infrastructure for minimizing compatibility issues among interacting components of different dialect versions, comprising:

a plurality of components located at nodes of said network and between which requests may be exchanged formatted according to a dialect implemented by a network protocol, each request being of a type and version from among a plurality of types and

versions and having a header carrying a type-version identifier indicating a corresponding type and version of said request; and

each said component including,

an input port for receiving one of said requests,

at least one handler supporting requests of a corresponding one of the plurality of types and versions,

a pointer array having a plurality of elements each being a pointer to a corresponding one of said handlers, and

switching logic operable to extract said type-version identifier carried by said received request, and use said extracted type-version identifier to index said pointer array to determine and invoke a selected one of said handlers.